Simulation du comportement d'un opérateur en situation de combat naval

Isabelle Toulgoat, Pierre Siegel

Simulation du comportement d'un opérateur en situation de combat naval

Isabelle Toulgoat[1], **Pierre Siegel**[2], **Yves Lacroix**[3]

[1]DCNS, [2]Université de Provence,[3]Laboratoire Systèmes Navals Compexes

isabelle.toulgoat@dcnsgroup.com, siegel@cmi.univ-mrs.fr, yves.lacroix@univ-tln.fr

Résumé

Cet article porte sur une application qui permet de simuler le comportement d'un officier dans un sous-marin. Nous nous sommes intéressés plus particulièrement à ses réactions en cas de détection d'un sous-marin adverse. Cette application a été implémentée en Prolog.

Dans les simulations de combat naval, les performances opérationnelles de navires militaires sont estimées pour un scénario donné. Dans les modèles courants, les réactions de l'opérateur sont prédéfinies. Ces réactions ne sont pas réalistes : la décision de l'opérateur peut conduire à des réactions inattendues. Cet article présente une méthode pour modéliser le comportement d'un opérateur dans les simulations. Cette méthode permet de raisonner sur des informations incertaines et révisables : un opérateur a une vue partielle de son environnement et il doit réviser ses décisions avec l'arrivée et le changement d'informations. Notre méthode utilise une logique non-monotone : les règles de comportement sont formalisées avec la logique des défauts, à laquelle nous ajoutons une gestion du temps. Nous utilisons des préférences pour gérer le choix entre plusieurs règles, avec une technique simple de probabilité.

Cette méthode a permis de modéliser les réactions d'un officier dans un scénario faisant intervenir deux sous-marins adverses. Cette application, implémentée en Prolog, a été interfacée avec un atelier de simulation de combat naval de DCNS.

Abstract

This article deals with a study case which models the officer behavior in a submarine, and more especially in a case of an adversary detection. This application has been implemented in Prolog.

Naval action's simulations estimate the operational performance of warships or submarines for a given scenario. In common models, the operator's reactions are predefined. This is not realistic : the operator's decision can produce unexpected reactions.

This article presents a method to model operator decision in simulations. This method allows to reason about incomplete, revisable and uncertain information : an operator has only partial information about his environment and must revise his decisions. Our method uses a non-monotonic logic : the rules of behavior are formalized with the default logic, to which we added a consideration of time. Our method uses preferences to manage the choices between different rules, with simple probability techniques.

This application, implemented in Prolog, has been interfaced with DCNS simulator framework and applied to a scenario involving two adverse submarines.

1 Introduction

Le but des simulations de combat naval est d'évaluer les performances opérationnelles de navires militaires ou de forces navales dans un scénario et un théâtre d'opérations données. L'atelier de simulation de DCNS, ATANOR, permet de modéliser des navires et des équipements du système de combat et de dérouler un scénario en temps simulé en faisant interagir les différents modèles [13].

Dans cet atelier de simulation, le comportement est modélisé avec des réseaux de Pétri [8]. Un réseau de Pétri est constitué de places, qui modélisent les différents états de l'équipement et de transitions entre ces places. Les transitions entre ces places sont activées par des évènements internes et externes. Une marque détermine la place courante du réseau de Pétri à un instant donné. Une seule place peut être active à la fois, ce qui interdit les fonctionnements parallèles au sein de l'équipement.

La modélisation des règles de comportement avec un réseau de Pétri donne des réactions prédéfinies et automatiques. Or, dans une situation tactique complexe, l'analyse et la décision d'un opérateur peut conduire à une réaction inattendue. De plus, un inconvénient de la modélisation par réseau de Pétri est de recommencer à implémenter une nou-

velle machine à états si l'on veut ajouter une nouvelle action [5].

Le but de ce travail est de développer un système qui permet de modéliser les lois de comportement d'un opérateur dans les simulations de performances. Nous avons travaillé sur un cas d'application faisant intervenir deux sous-marins adverses, et plus spécialement sur les réactions de l'officier en cas de détection de son adversaire. Un des objectifs est de pouvoir interfacer ce système de modélisation du comportement avec l'atelier de simulation ATANOR : après avoir récupérer les données d'ATANOR sur le sous-marin et son environnement, le système doit pouvoir renvoyer les instructions de manoeuvres pour le sous-marin à ATANOR.

Ce système doit répondre à plusieurs exigences :

- permettre de modéliser les lois de comportement de l'opérateur.
- permettre de raisonner sur des informations incomplètes, révisables et incertaines. En effet, un opérateur a seulement une vue partielle de son environnement. Dans le cas d'un sous-marin, aveugle en immersion, les seules informations proviennent des sonars passifs. Ces informations sont incertaines et incomplètes : le sous-marin peut facilement perdre la détection, l'officier a seulement accès à une estimation de la trajectoire de son adversaire .... Les décisions de l'officier doivent pouvoir être révisées avec l'arrivée de nouvelles informations [3] [11].
- choisir entre plusieurs propositions lorsqu'il propose plusieurs actions possibles pour une même situation.
- permettre l'addition de nouvelles règles de comportement, sans avoir à modifier la représentation des connaissances et sans avoir à remettre en question les règles précédemment établies (à l'inverse des réseaux de Pétri dans lesquels les modifications sont compliquées).
- être capable de raisonner sur des règles générales, sans avoir à compiler de manière très précise toutes les informations. Il n'est pas nécessaire pour l'utilisateur de décrire tous les cas de figures.

Ce travail est demandé par la société DCNS pour des applications militaires : nous avons besoin d'un programme simple et robuste. Nous utilisons pour cela la logique non-monotone la plus connue : la logique des défauts. Nous y ajoutons une prise en compte du temps : nous avons les données sur le sous-marin à l'instant t, et nous en déduisons les instructions au temps suivant $t + 1$. Nous utilisons les préférences pour choisir entre les différentes actions, avec une technique simple de probabilités. Ce travail a été implémenté en Prolog et interfacé avec l'atelier de simulation ATANOR. Une grande partie de ce travail a été présenté à la conférence NMR [14].

Dans cet article, nous présenterons le cas d'application avec quelques règles de comportement. Ensuite, nous présenterons les limites de la logique classique et pourquoi nous utilisons une logique non-monotone. Nous expliquerons la formalisation des règles en logique des défauts. Nous utilisons seulement des défauts normaux et des clauses de Horn dans le but d'avoir un programme simple. Cependant, nous pouvons étendre ce travail à d'autre cas d'application avec des règles plus compliquées. Ensuite, nous expliquons le choix entre les extensions grâce aux préférences et à une technique simple de probabilité. Finalement, nous présentons l'interface réalisée avec ATANOR et des résultats.

2 Cas d'application : détection et pistage d'un sous-marin

Dans un scénario faisant intervenir deux sous-marins, nous modélisons la décision d'un officier de quart dans un sous-marin, en fonction des évènement perçus sur la situation tactique. Nous avons défini des règles de comportement en interrogeant des sous-mariniers.

Voici quelque exemples de règles :

- Règle 1 : Tant que le sous-marin n'a pas de détection, il poursuit une trajectoire aléatoire de recherche dans sa zone de patrouille. Une trajectoire aléatoire de recherche est définie de la façon suivante : le sous-marin avance tout droit et, de temps en temps, il change de cap. Le sous-marin est sourd dans son baffle (à l'arrière du sous-marin, la réception des sonars est amoindrie pour diverses raisons), cette manoeuvre lui permet de vérifier qu'il n'est pas pisté. Le but de cette manoeuvre est de quadriller le plus largement possible la zone, afin d'avoir le plus de chance de détecter un intrus.

  Remarque : c'est une règle de changement minimal [3] [6] [17] . Cette règle est appliquée tant que le sous-marin n'a pas de nouvelles informations.
- Règle 2 : Si le sous-marin détecte un autre sous-marin, l'officier engage les actions suivantes :
  - Manoeuvre d'anti-collision : il manoeuvre de façon à mettre l'ennemi dans un certain gisement (angle entre la direction de détection du but et la ligne de foi du sous-marin qui détecte).
  - Manoeuvre d'élaboration de la solution : il manoeuvre dans le but de confirmer ses informations sur la trajectoire de l'ennemi.
  - Ralliement du poste de pistage : une fois que l'officier s'est assuré que le but ne l'a pas détecté, il peut rallier son baffle, position dans laquelle il ne sera pas détecté.
  - Pistage : une fois que le sous-marin se trouve dans le baffle du but, il peut commencer le pistage. Pour cela, il effectue des tronçons tout en restant dans la baffle.
- Règle 3 : Si les deux sous-marins sont trop proches,

l'officier doit effectuer un dérobement. Il s'éloigne afin de ne pas être contre-détecté et de garder une distance de sécurité.

- Règle 4 : Si le sous-marin est un sous-marin diesel et que quelques heures se sont écoulées depuis la dernière charge de batterie, il doit se mettre au schnorchel. Pour cela, il remonte à l'immersion periscopique et utilise son tube d'air.
- Règle 5 : Si le sous-marin perd la détection, l'officier rallie la dernière position connue de son adversaire. Si il le retrouve, il peut reprendre ses actions de pistage. Sinon, il reprend une trajectoire aléatoire.
- Règle 6 : Avec le sonar à évitement d'obstacle MOAS (Mine and Obstacle Avoidance System), le sous-marin peut détecter des mines, des gros rochers ou encore des falaises. Si le sous-marin détecte un gros rocher, il change de cap afin de placer le rocher de côté.

Ces règles peuvent être en compétition : à un même moment, le sous-marin peut avoir besoin d'effectuer plusieurs actions. Par exemple, il peut avoir besoin de se mettre au schnorchel et également de poursuivre son action de pistage. L'officier doit alors gérer ces choix.

3 La logique classique et ses limites

Les logiques classiques, comme la logique mathématique ou encore le calcul propositionnel, sont des logiques monotones : si on ajoute une information ou une formule E' à une formule E, tout ce qui était déductible de E sera déductible à partir de E $\cup$ E'. Cette propriété de monotonie va être gênante si l'on veut raisonner sur des informations incomplètes, incertaines et révisables. En effet, dans ce cadre, il sera courant d'invalider des conclusions précédemment établies lorsque l'on ajoute de nouvelles informations ou lorsque les informations sont changées [15].

Par exemple, prenons la règle : "Généralement, un sous-marin, qui n'a pas de détection, fait une trajectoire aléatoire de recherche". A première vue, on peut exprimer ce type d'information avec la logique du premier ordre :
Règle 1 :

$\forall x, \neg detection(x) \rightarrow trajectoire\_aleatoire(x)$

Cette formulation est cohérente si la seule information connue est "Le sous-marin n'a pas de détection". Mais si on ajoute la règle : "Si quelques heures se sont écoulées depuis la dernière charge de batterie, le sous-marin doit se mettre au schnorchel", elle est définie de la façon suivante :

Règle 2 : $\forall x, Tlc(x) \geq 5 \rightarrow schnorchel(x)$, avec Tlc le temps écoulé depuis la dernière charge et $schnorchel$ l'action de se mettre au schnorchel.

Il est difficile de gérer des règles générales avec un nombre important d'exceptions [11]. De plus, nous ne pouvons pas réviser le raisonnement et les conclusions. Sachant que le sous-marin n'a pas de détection, nous déduisons qu'il doit faire la trajectoire aléatoire. Mais si nous savons que quelques heures se sont écoulées depuis la dernière charge de batterie, nous en déduisons que le sous-marin doit se mettre au schnorchel. Nous obtenons deux conclusions qui ne sont pas consistantes : le sous-marin ne peut pas en même temps exécuter ces deux actions.

De plus, nous avons besoin d'une logique pour raisonner sur des informations incertaines. Dans le cas d'un sous-marin, aveugle en immersion, les seules informations viennent des sonars passifs, elles sont incertaines et incomplètes [9].

4 La logique non-monotone et la logique des défauts

Une logique non monotone permet d'éliminer la propriété de monotonie de la logique classique : si un raisonnement donne des conclusions avec une base de connaissance donnée, ces conclusions peuvent être revues avec l'arrivée de nouvelles connaissances. Cela permet de prendre en compte les informations incomplètes, incertaines et révisables. Cette logique présente une similitude naturelle avec le raisonnement humain : par manque d'information ou manque de temps, une personne peut raisonner avec des connaissances partielles et réviser les conclusions au besoin lorsqu'elle a plus d'informations.

La logique des défauts, introduite par Ray Reiter [10] en 1980, est la logique non-monotone la plus utilisée. Elle formalise le raisonnement par défaut : des conclusions sensées peuvent être faites en l'absence de preuve contraire. Une théorie des défauts $\Delta = (D, W)$ est composée d'un ensemble de faits W, qui sont des formules issues du calcul propositionnel ou bien de la logique du premier ordre et d'un ensemble de défauts D, qui sont des règles d'inférences à contenu spécifique. Les défauts permettent de gérer l'incertitude. Rappelons la définition d'un défaut et la définition d'une extension :

Définition : Défaut Un défaut est une expression de la forme : $\frac{A(X) : B(X)}{C(X)}$, où A(X), B(X) et C(X) sont des formules et X est un ensemble de variables.

A(X) est le prérequis, B(X) est la justification et C(X) est le conséquent.

Le défaut $\frac{A(X) : B(X)}{C(X)}$ signifie : si A(X) est vérifié, si il est possible que B(X) soit vrai (B(X) est consistant), alors C(X) est vrai.

Si $B(X) = C(X)$, le défaut est dit normal (la justification et le conséquent sont identiques). Les défauts normaux signifient " Les A sont des B, sauf exceptions ", "Normalement, les A sont B".

Définition : Extension L'utilisation des défauts augmente les formules déduites de la base de connaissances W. Pour cela, on calcule les extensions qui sont définies de la façon suivante :

E est une extension de Δ si et seulement si $E = \cup_{i=0,\infty} E_i$, avec

$$E_0 = W \text{ et pour } i \geq O,$$
$$E_{i+1} = Th(E_i) \cup \{C/(\frac{A:B}{C}) \in D, A \in E_i, \neg B \notin E\}$$

où $Th(E_i)$ désigne l'ensemble des théorèmes obtenus de façon monotone à partir de $E_i : Th(E_i) = \{w/E_i \vdash w\}$.

Il est important de noter que E apparaît dans la définition de E_{i+1}. Il n'est donc pas possible de construire E avec un algorithme incrémental.

Lorsque l'on travaille avec des défauts normaux, l'extension est définie de la façon suivante :

E est une extension de Δ si et seulement si $E = \cup_{i=0,\infty} E_i$, avec

$$E_0 = W \text{ et pour } i \geq O,$$
$$E_{i+1} = Th(E_i) \cup \{C/(\frac{A:C}{C}) \in D, A \in E_i, \neg C \notin E_i\}$$

où $Th(E_i)$ désigne l'ensemble des théorèmes obtenus de façon monotone à partir de $E_i : Th(E_i) = \{w/E_i \vdash w\}$.

Pour notre cas d'application, nous utilisons seulement des défauts normaux, mais nous pourrions étendre notre travail aux autres défauts.

5 Formalisation des règles avec la logique des défauts

5.1 Prise en compte du temps

Pour formaliser les règles de comportement, on utilise la logique des défauts, à laquelle on ajoute une prise en compte du temps. En effet, nous avons les données sur le sous-marin à l'instant t et nous déduisons les instructions pour le sous-marin au temps suivant $t + 1$, en fonction de l'état du sous-marin et des nouvelles informations. Pour introduire le temps, nous avons commencé par travailler avec l'article écrit par Cordier et Siegel [3]. Il traite du problème de révision : comment garder à jour une base de connaissances dynamique (qui évolue au court du temps) et comment réviser les informations.

Nous avons besoin de prendre le temps en considération dans les définitions des faits W et des défauts D de la logique des défauts $\Delta = (D, W)$ [12].

5.2 Définition des faits avec prise en compte du temps

L'ensemble des faits W est défini avec les formules de la logique propositionnelle ou la logique du premier ordre. Nous utilisons uniquement les clauses de Horn. Elles permettent d'écrire deux type de règles :

- les clauses de Horn avec un littéral positif :
  $(g(t) \vee \neg f_1(t) \vee \neg f_2(t) \vee \ldots \vee \neg f_k(t))$, où les $f_i(t)$ et $g(t)$ sont des littéraux positifs au temps t. Cette formule peut également s'écrire avec une implication : $(f_1(t) \wedge f_2(t) \wedge \ldots \wedge f_k(t)) \rightarrow g(t)$.
  Ce type de règles permet de définir les règles qui sont toujours vraies, ce sont les règles classiques des systèmes experts. Par exemple, nous formalisons la règle : " Si le sous-marin est en trajectoire aléatoire, il doit tourner d'une angle compris entre α et β", de la façon suivante :
  $trajectoire\_aleatoire(X_t) \rightarrow tourner(X_t, (\alpha, \beta))$
- les clauses de Horn sans littéral positif sont écrites de la façon suivante : $(\neg f_1(t) \vee \neg f_2(t) \ldots \vee \neg f_k(t))$, qui est équivalent à $\neg(f_1(t) \wedge f_2(t) \ldots \wedge f_k(t))$. Nous utilisons ces règles pour définir les exclusions mutuelles deux à deux, cela correspond aux prédicats qui ne peuvent pas être exécutés en même temps :
  $(\neg f_1(t) \vee \neg f_2(t))$, qui est équivalent à $\neg(f_1(t) \wedge f_2(t))$.
  Nous pouvons définir une règle telle que : "Le sous-marin ne peut pas faire en même temps une trajectoire aléatoire et aller au schnorchel" : $\neg(trajectoire\_aleatoire(X_t) \wedge schnorchel(X_t))$.

5.3 Définition des défauts avec prise en compte du temps

Les défauts D sont des règles d'inférence à contenu spécifique, qui prennent en compte l'incertitude. Ils expriment le fait que, si il n'y a pas de contradiction à exécuter une action, le sous-marin peut le faire. Nous utilisons seulement les défauts normaux. Ils permettent de formaliser des règles telles que : "Si le sous-marin n'a pas de détection, il fait une trajectoire aléatoire", de la façon suivante :

$$\frac{\neg detection(X_t) : trajectoire\_aleatoire(X_{t+1})}{trajectoire\_aleatoire(X_{t+1})}.$$

Ce défaut signifie : "Si le sous-marin n'a pas de détection au temps t et si il lui est possible de faire une trajectoire aléatoire au temps $t + 1$, il fait une trajectoire aléatoire au temps $t + 1$".

Les défauts nous permettent de définir les comportements généraux du sous-marin (se mettre au schnorchel, éviter la collision, pister ...). Ensuite, l'ensemble des faits permet de spécifier, pour chaque comportement, l'action à réaliser (changement de cap, vitesse, immersion) ainsi que que les exculsions mutuelles.

5.4 Le calcul d'extension

Nous utilisons le calcul d'extension pour étudier tous les défauts et retenir ceux qui répondent au problème. Chaque

extension est une manoeuvre possible pour le sous-marin : en fonction de l'état du sous-marin au temps t, une extension donne une manoeuvre possible pour le sous-marin au temps $t + 1$. Les défauts normaux permettent d'assurer l'existence d'au moins une extension. Généralement, nous aurons plusieurs extensions pour une même base de connaissance.

Nous pourrions utiliser les ASP (Answer set programming) [7] pour calculer les extensions. Afin de garder un programme simple, nous avons préféré implémenter notre propre calcul d'extension. Les défauts normaux et les clauses de Horn nous permettent d'implémenter de façon simple le calcul des extensions avec le langage Prolog. Nous appelons notre programme NoMROD, pour Non-Monotonic Reasoning for Operator Decision.

6 Sélection des extensions avec des préférences

Le but de cette partie est de simuler la décision de l'officier. Dans une situation tactique, la décision d'un opérateur est un aspect clé. A chaque instant, l'officier doit choisir une action. Nous définissons une méthode qui permet de choisir entre les différentes extensions. Cette méthode permet de simuler différents types de comportements, qui dépendent du caractère de l'officier.

Cette méthode de sélection a été définie en trois étapes :

1. la définition de principes généraux auxquels doit répondre la sélection de l'extension.
2. la définition d'une fonction de poids pour chaque extension. Cette fonction de poids prend en compte le comportement de l'officier à deux niveaux :
   - Niveau 1 : l'importance de chaque défaut en fonction de différents critères. Pour cela, on définit des coefficients de préférences $C_1, \ldots, C_n$, qui permettent de définir des préférences en fonction des différents critères. Ces coefficients décrivent l'importance de chaque action.
   - Niveau 2 : le caractère de l'officier avec des coefficients de caractères $\beta_1, \ldots, \beta_n$. Ces coefficients définissent l'importance que l'officier accorde à chaque critère. Grâce à ces coefficients de caractères, nous pouvons modéliser différents officiers : prudent, téméraire ...

   Les principes de sélection de l'extension ainsi que la fonction de poids ont été développés pour notre cas d'application mais peuvent être facilement généralisés.
3. la définition d'une méthode pour sélectionner une extension, en ajoutant une part d'aléatoire dans le choix de l'extension.

6.1 Les principes de sélection d'une extension

Premièrement, nous étudions des principes et les exigences auxquels la sélection de l'extension doit répondre :

1. choisir les extensions les plus intéressantes et laisser les autres de côté.
   Exemple : NoMROD propose deux extensions :
   – faire une trajectoire aléatoire de recherche.
   – aller au schnorchel.
   La trajectoire aléatoire de recherche est une règle de changement minimal : elle est appliquée tant que le sous-marin n'a pas de nouvelles informations. L'officier choisira donc d'aller au schnorchel.
2. choisir les extensions obligatoires pour la survie de l'équipage.
   Par exemple, l'officier peut repousser la montée au schnorchel, si il est en train d'exécuter une autre action importante (par exemple le pistage). Lorsque cette règle est vérifiée, l'officier a approximativement trente minutes de batterie. Quand cette réserve sera pratiquement vide, l'officier sera obligé d'aller au schnorchel.
3. gérer les choix entre plusieurs extensions.
   Exemple : NoMROD propose deux extensions :
   – éviter la collision avec un sous-marin.
   – éviter la collision avec un gros rocher.
   Ces deux comportements sont très importants pour la sauvegarde du sous-marin. NoMROD doit être capable de choisir entre les extensions de même importance.
4. respecter le changement minimal : tant que le sous-marin n'a pas de nouvelles informations, il reste dans le même état, il ne change pas de comportement. On doit donner plus d'importance à une action déjà commencée.
   Avec cette règle, l'officier persistera dans ses choix, il n'oscillera pas entre plusieurs comportements. Cependant, NoMROD doit être capable de stopper une action si une autre devient obligatoire.
   Example : NoMROD propose deux extensions :
   – pister l'ennemi.
   – monter au schnorchel.
   Supposons que le système choisisse le pistage. Ces deux extensions seront proposées tant que la montée au schnorchel n'aura pas été effectuée. Le meilleur choix pour l'officier est de poursuivre le pistage, et lorsque le schnorchel devient obligatoire (le sous-marin a juste assez de batterie pour monter à la surface), l'officier doit effectuer cette action.
5. le sous-marin ennemi ne doit pas deviner quelle action notre officier choisira.

Ces principes sont des principes de bon sens, qui sont facilement généralisables à d'autre cas d'application.

6.2 La fonction de poids pour les extensions

Nous définissons une fonction de poids pour donner des poids aux extensions. Ces poids quantifient l'importance des extensions. Nous utilisons une méthode d'aide à la décision multi-critères, qui sert à modéliser les préférences d'un décideur. Ces méthodes servent à résoudre des problèmes décisonnels complexes, où plusieurs critères doivent être pris en compte dans le choix [1],[16]. Il existe plusieurs catégories de méthodes d'aide à la décision multi-critères. Nous utilisons la théorie de l'utilité multi-attribut, qui permet d'agréger les différents points de vue en une fonction unique. Elle est basée sur l'axiome suivant :

Tout décideur essaye inconsciemment de maximiser une fonction $U = U(g_1, \ldots, g_n)$, qui agrège tous les points de vue à prendre en compte (avec g_i les critères).

Nous utilisons une fonction d'aggrégation simple : une somme pondérée.

Chaque extension utilise des défauts. Pour quantifier l'importance des extensions, nous introduisons des coefficients de préférences sur les défauts.

6.2.1 Les coefficients de préférences sur les défauts

Chaque défaut est un comportement général. Pour spécifier l'importance des défauts, nous leurs attribuons des coefficients de préférences.

De façon similaire, les préférences sont utilisées dans des systèmes de raisonnement non-monotone. Par exemple, Brewka [2] a défini une logique des défauts avec des priorités et une définition d'ordre dans lequel les défauts doivent être appliqués. Delgrande et al [4] ont présenté une classification des préférences pour les approches basées sur des raisonnements non-monotones.

Nous définissons différents coefficients de préférences $C_1, \ldots, C_n$ pour spécifier l'importance des défauts en fonction de différents critères. Par exemple, nous pouvons spécifier l'importance des défauts pour la sauvegarde du sous-marin, pour l'efficacité dans la mission du sous-marin, pour l'obéissance aux ordres, pour des critères écologiques ...

Pour chaque défaut D_j, nous attribuons les valeurs de ces coefficients $C_{1j} \ldots C_{kj}, \ldots, C_{nj}$. Le coefficient C_{kj} définit l'importance du défaut j par rapport au critère k. Nous fixons arbitrairement ces coefficients entre 0 et 1000.

Par exemple, on définit quatre défauts : $\{D_1, D_2, D_3, D_4\}$, et deux coefficents : la sauvegarde du sous-marin $C_{sauvegarde}$ et l'efficacité dans la mission du sous-marin $C_{efficacite}$. Pour chaque défaut, on définit la valeur des coefficients de sauvegarde et d'efficacité (cf tableau 1).

TAB. 1 – Valeurs des coefficients pour chaque défaut.

Défauts	$C_{sauvegarde}$	$C_{efficacite}$
D_1	500	600
D_2	10	800
D_3	1000	900
D_4	50	60

TAB. 2 – Scores des extensions.

Scores	E_1	E_2
$Score_{sauvegarde}$	550	1060
$Score_{efficacite}$	660	1760

6.2.2 Fonction d'utilité

Chaque extension E_i utilise des défauts : $E_i = \{D_1 \ldots, D_j, \ldots, D_m\}$. Dans notre cas, chaque extension utilise au moins un défaut.

Pour chaque extension E_i, nous calculons les scores $\{Score_1(E_i), \ldots, Score_k(E_i), \ldots, Score_n(E_i)\}$, qui correspondent respectivement aux critères $1, \ldots, n$.

Un score $Score_k(E_i)$ est la somme des coefficients C_k de chaque défaut utilisé dans l'extension E_i : $Score_k(E_i) = \sum_{j=1}^{m} C_{kj}$.

Nous supposons que NoMROD propose p extensions. Pour chaque $Score_k(E_i)$ pour une extension E_i, nous calculons une fonction d'utilité $\mu_k(E_i)$. Nous voulons que la somme des fonctions d'utilité μ_k soit égale à 1 : $\sum_{j=1}^{p} \mu_k(E_j) = 1$. Le score $Score_k(E_i)$ est divisé par la somme de tous les $Score_k(E_j)$ des p extensions proposées par le système : $\mu_k(E_i) = \dfrac{Score_k(E_i)}{\sum_{j=1}^{p} Score_k(E_j)}$.

Reprenons les défauts donnés en exemple dans le paragraphe 6.2.1.

On suppose que l'on doit choisir entre deux extensions : $E_1 = \{D_1, D_4\}$ et $E_2 = \{D_2, D_3, D_4\}$.

On calcule les scores des extensions (cf tableau 2) et les fonctions d'utilité (cf tableau 3).

6.2.3 Le caractère de l'officier

Nous voulons modéliser le caractère de l'officier. En fonction de son caractère, l'officier adoptera des tactiques différentes. Par exemple, un officier prudent donnera plus d'importance aux comportements qui assurent la sauvegarde du sous-marin. Un officier téméraire favorisera les comportements importants pour la réussite de la mission

TAB. 3 – Fonctions d'utilité.

μ	E_1	E_2
$\mu_{sauvegarde}$	$\frac{550}{1610}$	$\frac{1060}{1610}$
$\mu_{efficacit}$	$\frac{660}{2420}$	$\frac{1760}{2420}$

du sous-marin.

Pour modéliser ces caractères, nous définissons des coefficients de caratères $\beta_1, \ldots, \beta_n$, de telle façon que la somme de ces coefficients soit égale à 1 : $\sum_{k=1}^{n} \beta_k = 1$. Le coefficient β_k correspond à l'importance accordé par l'officier au critère k.

Définissons, par exemple, un officier plutôt prudent. En reprenant l'exemple du paragraphe 6.2.1, on doit attribuer deux coefficients de caractère : un coefficient pour la sauvegarde $\beta_{sauvegarde}$ et un coefficient pour l'efficacité $\beta_{efficacite}$. Un officier prudent va préférer favoriser la sauvegarde de son sous-marin plutôt que l'efficacité dans la réussite de la mission, prenons par exemple : $\beta_{sauvegarde} = 0.6$ et $\beta_{efficacite} = 0.4$.

6.2.4 Fonction de poids

Finalement, nous obtenons la fonction de poids de chaque extension E_i, en faisant la somme des fonctions d'utilité μ_k, pondérées par les coefficients de caractères β_k :

$$P(E_i) = \sum_{k=1}^{n} \beta_k \mu_k(E_i).$$

La somme des fonctions de poids des extensions a la propriété suivante : $\sum_{i=1}^{p} P(E_i) = 1$

En reprenant l'exemple précédent, on obtient les fonctions de poids suivantes :

$P(E_1) = \beta_{sauvegarde} * \mu_{sauvegarde}(E_1) + \beta_{efficacit} * \mu_{efficacit}(E_1)$

$P(E_2) = \beta_{sauvegarde} * \mu_{sauvegarde}(E_2) + \beta_{efficacit} * \mu_{efficacit}(E_2)$

Finalement, on a $P(E_1) = 0.31$ et $P(E_2) = 0.69$.

6.3 Choix aléatoire d'une extension

Dans une situation tactique, la décision d'un opérateur est un aspect clé, qui peut conduire à des réactions inattendues. En effet, chaque opérateur possède ses propres réactions. Dans notre cas d'application, le choix ne doit donc pas toujours être déterministe.

De plus, avec des réactions attendues, le sous-marin ennemi pourrait facilement deviner les réactions de l'officier.

Nous avons donc besoin d'une méthode qui prend en compte les réactions inattendues. Pour ces raisons, nous ne choisissons pas toujours les extensions avec un poids maximum. Nous préférerons introduire d'avantage d'incertitude avec un choix aléatoire. Cependant, ce choix aléatoire doit être cohérent avec les principes qui guident la décision de l'officier et avec les principes de sélection d'une extension définis au paragraphe 6.1.

6.3.1 Choix aléatoire

Le choix aléatoire est basé sur un tirage aléatoire : on a p extensions : $E_1, \ldots, E_p$, et les fonctions de poids respectives : $P(E_1), \ldots, P(E_p)$, telles que $\sum_{i=1}^{p} P(E_i) = 1$. Chaque fonction de poids $P(E_i)$ correspond à la probabilité pour l'extension d'être choisie.

Exemple 1 : Nous devons choisir entre deux extensions :
- E_1 avec la fonction de poids $P(E_1) = 0.1$.
- E_2 avec la fonction de poids $P(E_2) = 0.9$.

Nous avons une chance sur dix de choisir l'extension E_1, et neuf chances sur dix de choisir l'extension E_2. Avec le choix aléatoire, nous pouvons gérer les choix entre plusieurs extensions (principe 3).

6.3.2 Correction de la fonction de choix des extensions

Le tirage aléatoire est réaliste si nous avons à choisi entre des extensions qui ont des poids proches. Par contre, il nous pose des problèmes dans certains cas. En effet, si nous avons une extension avec un poids très important, il semble plus logique de choisir celle là (exemple 1). Nous avons le même problème dans le cas suivant (exemple 2). NoMROD propose six extensions : cinq extensions avec une même fonction de poids à 0.1 et une extension avec un poids de 0.5. Le tirage aléatoire donne autant de chance d'être choisies aux cinq extensions avec la fonction de poids à 0.1 : $5 * 0.1 = 0.5$, qu'à l'extension avec la fonction de poids à 0.5.

Nous devons modifier la fonction de poids, dans le but de donner un poids plus important aux extensions importantes et un poids moins important aux autres. Pour cela, nous appliquons une correction aux fonctions de poids : la fonction puissance $f(x) = x^k$, avec $k > 1$. La correction est appliquée de la façon suivante :

- Nous devons choisir entre p extensions : $E_1, \ldots, E_p$, avec les fonctions de poids respectives : $P(E_1), \ldots, P(E_p)$, telles que $\sum_{i=1}^{p} P(E_i) = 1$.
- Nous appliquons la fonction puissance $f(x) = x^k$, avec $k > 1$: $P(E_1)^k, \ldots, P(E_p)^k$. Plus la puissance

k est importante, plus les extensions avec des petits poids seront réduites.

– La somme des fonctions de poids est ensuite ramenée à 1 : nous divisons par la somme des fonctions de poids de toutes les extensions $\frac{P(E_j)^k}{\sum_{i=1}^{p} P(E_i)^k}$.

Cette correction donne plus d'importance aux extensions avec des fonctions de poids importantes, et moins d'importance aux autres. Pour le moment, nous ne fixons par les valeurs de la puissance k, nous voulons tester différentes valeurs.

Appliquons cette correction à l'exemple 2. Nous prenons la fonction puissance $f(x) = x^2$. La fonction de poids 0.1 devient $0.1^2 = 0.01$ et la fonction de poids à 0.5 devient $0.5^2 = 0.25$.

On ramène la somme des fonctions de poids à 1. La somme des fonctions de poids avec correction est $\sum_{i=1}^{6} P(E_i) = 0.3$.

On obtient cinq extensions avec la fonction de poids $\frac{0.1^2}{0.3} = 0.04$ et une extension avec la fonction de poids $\frac{0.5^2}{0.3} = 0.8$. Avec la correction, on donne plus de chance à l'extension avec un poids important d'être choisi.

6.3.3 Filtrage des extensions avec de fonctions de poids trop faibles

Pour être sûr de choisir l'extension la plus intéressante et de laisser les autres de côté (principe 1), nous éliminons les extensions avec des fonctions de poids trop petites. On fixe un seuil : les extensions avec une fonction de poids inférieure à ce seuil sont supprimées. Pour le moment, nous ne fixons pas cette valeur de seuil : nous voulons tester différentes valeurs (nous pourrions par exemple fixer cette valeur en fonction de la fonction de poids maximale).

Maintenant que nous avons défini les fonctions de poids pour les extensions, nous définissons une règle qui prend en compte le changement minimal.

6.4 Prise en compte du changement minimal

Pour respecter le changement minimal (principe 4), nous définissons une règle générale de changement minimal. Pour cela, on garde en mémoire le comportement du sous-marin en cours (trajectoire aléatoire, l'évitement de la collision, le pistage, ...) au temps t. On appelle ce comportement $Comportement(t)$. Chaque comportement est le conséquent d'un défaut D_i :

$$D_i = \frac{Prerequis(t) : Comportement(t+1)}{Comportement(t+1)}.$$

Avec la règle du changement minimal, nous voulons donner plus de chance à une action déjà commencée. Pour cela, nous définissons le défaut suivant :

$$D_{ch\_min} = \frac{Comportement(t) \wedge Prerequis(t) : Comportement(t+1)}{Comportement(t+1)}.$$

Cette règle signifie : " Si le prérequis du comportement précédent est toujours vrai au temps t, et si il est possible de rester dans ce comportement au temps $t+1$, le sous-marin peut garder le même comportement au temps $t+1$".

Cette règle donne plus de chance à une action déjà commencée et permet à l'officier de persister dans ses choix.

7 Interface avec l'atelier de simulation et résultats

Une interface a été réalisée entre NoMROD et l'atelier de simulation ATANOR. ATANOR envoie à NoMROD les informations sur :

– le sous-marin auquel on applique les règles de comportement (cap, vitesse, immersion, position, ...)
– le sous-marin ennemi (détection, position estimée, vitesse estimée, ...).

NoMROD compile les règles de comportement, sélectionne une extension et renvoie les instructions de cap, vitesse et immersion à ATANOR.

Sur la figure 1, nous avons une exécution de NoMROD, interfacée avec l'atelier de simulation ATANOR. Cette exécution simule un scénario d'un peu moins de trois heures.

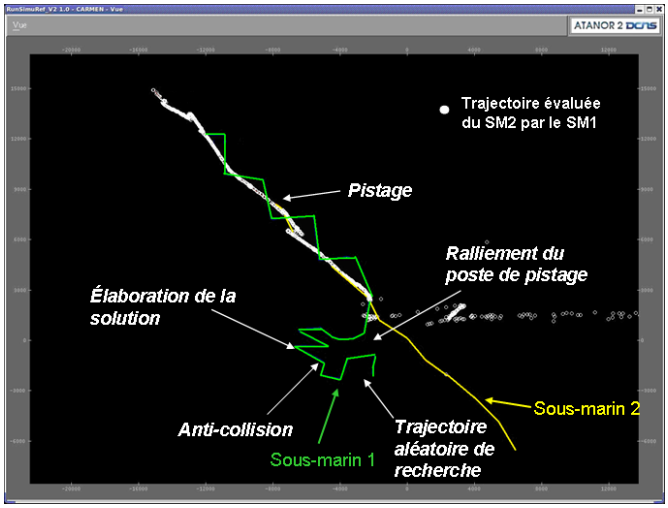

FIG. 1 – Détection et pistage d'un sous-marin

On appelle Sous-marin 1, le sous-marin auquel on applique les règles de comportement de NoMROD. Et on note Sous-marin 2 le sous-marin ennemi, dont le comportement est défini par ATANOR. Le but du sous-marin ennemi est de traverser la zone de patrouille sans être détecté.

Dans ce scénario, le sous-marin 1 fait une trajectoire aléatoire de recherche, car il n'a pas de détection. Lorsqu'il détecte l'ennemi, l'officier enclenche les actions suivantes : évitement de la collision, élaboration de la solution, ralliement du poste de pistage, pistage. Au début, la trajectoire évaluée du sous-marin 2 par le sous-marin 1 est éloignée de la trajectoire réelle. La manoeuvre d'élaboration de la solution permet d'obtenir une meilleure estimation de la trajectoire du sous-marin ennemi.

Nous obtenons un programme efficace. Pour simuler un scénario d'environ 2h40, le temps de calcul utilisé par ATANOR et NoMROD est d'environ 6 secondes et le programme NoMROD ne prend que 20 % de ce temps de calcul.

Ce scénario a été validé par des sous-mariniers, qui retrouvent bien les actions qu'ils engagent en cas de détection d'un sous-marin ennemi.

8 Conclusion

La logique des défauts permet de formaliser les règles de comportement d'un officier dans un sous-marin, en prenant en compte les informations incomplètes, incertaines et révisables. La formalisation en logique des défauts permet d'écrire des règles générales, les défauts, sans avoir à prendre en compte les règles précédemment établies. Nous avons juste à prendre en compte l'écriture des exclusions mutuelles entre les différents comportements.

Le système obtenu compile les informations valables et donne toutes les possibilités d'action avec les extensions.

Pour simuler le choix de l'officier, nous avons défini des fonctions de poids pour chaque extension avec des coefficients de préférence sur les défauts et des coefficients de caractère. Nous ajoutons ensuite une part supplémentaire d'aléatoire dans le choix des extensions.

Nous devons maintenant travailler sur une méthode générale pour attribuer la valeur des coefficients de préférence sur les défauts et les coefficients de caractères (par exemple par apprentissage). Nous voudrions également tester d'autres fonctions de poids pour les extensions (par exemple l'intégrale de Choquet).

Références

[1] S. Ben Mena. Introduction aux méthodes multicritères d'aide à la décision. *Biotchnol.Agron.Soc.envion.*, pages 83–93, 2000.

[2] G. Brewka. Adding priorities and specificity to default logic. In L.Pereira and D.Pearce, editors, *Lecture Notes in Artificial Intelligence*, pages 247–260. European Workshop on Logics in Artificial Intelligence (JELIA'94), 1994.

[3] M.O. Cordier and P. Siegel. A temporal revision model for reasoning about world change. In *Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 732–739, 1992.

[4] J. Delgrande, T. Shaub, and H. Tompits. A classification and survey of preference handling approaches in non-monotonic reasoning. *Computational Intelligence*, 20(2) :308–334, 2004.

[5] Ferber. *Les systèmes multi-agents. Vers une intelligence collective.* InterEditions, 1995.

[6] M. Ginsberg and D.E Smith. Reasoning about action 1 : a possible worlds approach. *Readings in Non Monotonic Reasoning*, 1987.

[7] P. Nicolas, L. Garcia, and I. Stephan. Possibilistic stable models. In *International Joint Conferences on Artificial Intelligence*, 2005.

[8] C.A. Petri. Fundamentals of a theory of asynchronous information flow. In *1st IFIP World Computer Congress*, 1962.

[9] Jr. Prouty. *Displaying uncertainty : a comparison between submarine subject matter experts*. PhD thesis, Naval postgraduate school, Monterey, California, 2007.

[10] R. Reiter. A logic for default reasoning. *Artificial intelligence*, 1980.

[11] L. Sombé. *Raisonnement sur des informations incomplètes en intelligence artificielle*. Teknea, 1989.

[12] I. Toulgoat. Modélisation du processus de décision d'un opérateur dans les simulations de combat naval. In *Journées d'Intelligence Artificielle Fondamentales, Session doctorants,Marseille*, 2009.

[13] I. Toulgoat, J. Botto, Y. De lassus, and C Audoly. Modeling operator decision in underwater warfare performance simulations. In *Conference UDT, Cannes.*, 2009.

[14] I. Toulgoat, P. Siegel, Y. Lacroix, and J. Botto. Operator decision in naval action's simulations. In *13th International Workshop on Non-Monotonic Reasoning*, 2010.

[15] I. Toulgoat, P. Siegel, Y. Lacroix, and J. Botto. Operator decision modeling in a submarine. In *9th International Conference on Computer and IT Applications in the Maritime Industries*, pages 65–75, 2010.

[16] P. Vincke. *L'aide multicritère à la décision*. Ellipses, 1989.

[17] M. Winslett. Reasoning about actions using a possible model approach. In *Proceedings of the 7th National Conference of AI*, pages 89–93, 1988.